(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 605 868 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **93120933.2**

(51) Int. Cl.5: **G06F 9/38**

(22) Date of filing: **27.12.93**

A request for addition on page 1, lines 2-4-7-10 and 12 of the description with the references of the co-pending related European patent applications has been filed pursuant to Rule 88 EPC. A decision on the request will be taken during the proceedings before the Examining Division (Guidelines for Examination in the EPO, A-V, 2.2).

(30) Priority: **08.01.93 US 1872**

(43) Date of publication of application:
13.07.94 Bulletin 94/28

(84) Designated Contracting States:
AT BE CH DE ES FR GB IT LI NL SE

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Kau, Chin-Cheng**
6000 Whistlestop Cove
Austin, Texas 78749(US)
Inventor: **Ogden, Aubrey D.**
1905 Stonewreath Drive
Round Roch, Texas 78681(US)
Inventor: **Waldecker, Donald E.**
3306 Arrowhead Circle
Round Rock, Texas 78681(US)

(74) Representative: **Bravi, Alfredo**
c/o IBM SEMEA S.p.A
Direzione Brevetti
MI SEG 024
P.O. Box 137
I-20090 Segrate (Milan) (IT)

(54) Method and system for indexing the assignment of intermediate storage buffers in a superscalar processor system.

(57) A method and system for enhanced instruction dispatch efficiency in a superscalar processor system having intermediate storage buffers (60), general purpose registers (62), and a storage buffer index (58). A particular storage buffer (60) is assigned to a destination operand within a selected multiple scalar instruction. A relationship between the particular intermediate storage buffer (60) and a designated general purpose register (62) is stored in the storage buffer index (58) when the instruction which has been dispatched is replaced in the dispatcher by another instruction. Results of execution from the selected multiple scalar instruction are stored in the particular intermediate storage buffer (60) when the selected instruction is executed. The storage buffer index (58) is used to determine which storage buffers (60) to use as source operands for those instructions which are dispatched between the time that a storage buffer (58) has been assigned for a specific general purpose register (62) and the results of execution are moved from the storage buffer (60) into the general purpose register (62).

EP 0 605 868 A1

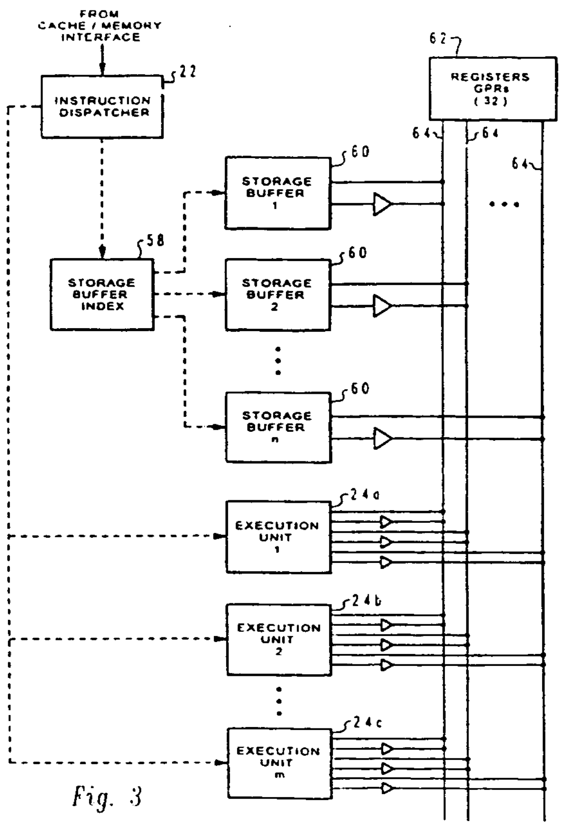
FROM
CACHE / MEMORY
INTERFACE
22
INSTRUCTION
DISPATCHER
62
REGISTERS
GPRs
(32)
64
64
64
60
STORAGE
BUFFER
1
58
STORAGE
BUFFER
INDEX
60
STORAGE
BUFFER
2
60
STORAGE
BUFFER
n
24a
EXECUTION
UNIT
1
24b
EXECUTION
UNIT
2
24c
EXECUTION
UNIT
m

Fig. 3

The present application is related to EP-A Serial No. ___________, entitled "Method and System for Selective Serialization of Instruction Processing in a Superscalar Processor System," EP-A Serial No. ___________, entitled "Method and System for Single Cycle Dispatch of Multiple Instructions in a Superscalar Processor System," EP-A Serial No. ___________, entitled "Method and System for Enhanced Instruction Dispatch in a Superscalar Processor System Utilizing Independently Accessed Intermediate Storage," EP-A Serial No. ___________, entitled "Method and System for Nonsequential Instruction Dispatch and Execution in a Superscalar Processor System," and EP-A Serial No. ___________, entitled "Method and System for Tracking Scalar Instructions Within a Superscalar Processor System," all filed of even date herewith and assigned to the assignee herein, and incorporated by reference herein.

The present invention relates in general to an improved data processing system, and in particular to a method and system for enhanced instruction dispatch efficiency in a superscalar processor system. Still more particularly, the present invention relates to a method and system for indexing the assignment of intermediate storage buffers in a superscalar processor system.

Designers of modern state-of-the-art processing systems are continually attempting to enhance performance aspects of such systems. One technique for enhancing data processing efficiency is the achievement of short cycle times and a low Cycles-Per-Instruction (CPI) ratio. An example of the application of these techniques to an enhanced data processing system is the International Business Machines Corporation RISC System/6000 (RS/6000) computer. The RS/6000 system is designed to perform well in numerically intensive engineering and scientific applications as well as in multi-user, commercial environments. The RS/6000 processor employs a superscalar implementation, which means that multiple instructions are issued and executed simultaneously.

The simultaneous issuance and execution of multiple instructions requires independent functional units that can execute concurrently with a high instruction bandwidth. The RS/6000 system achieves this by utilizing separate branch, fixed point and floating point processing units which are pipelined in nature. In view of the pipelined nature of the processors in such systems, care must be taken to ensure that a result from a particular instruction which is necessary for execution of a subsequent instruction is obtained prior to dispatching the subsequent instruction. One technique for ensuring that such so-called "data dependency hazards" do not occur is the restriction of the dispatching of a particular instruction until such time as all preceding instructions have been dispatched. While this technique ensures that data dependency hazards will not occur, the performance penalty encountered utilizing this technique is substantial.

As a consequence, modern superscalar data processing systems often utilize a so-called "data dependency interlock circuit." Such circuits contain logic which operates in concert with instruction dispatch circuitry to ensure that an instruction is not dispatched until such time as a result from a preceding instruction which is necessary for correct execution of that instruction has been obtained. The amount of logic required to implement a data dependency interlock circuit is directly proportional to the number of source operands within each instruction. A source operand is a field within an instruction which is utilized to access an operand within a register file for utilization in execution of that instruction.

While data dependency interlock circuits permit data dependency hazards to be avoided without encountering the substantial performance penalty described above, the design and implementation of data dependency interlock circuits for instructions which include larger numbers of source and destination operands becomes quite complex. The dependency interlocks inhibit dispatch of dependent instructions which prevent following instructions, which may be independent and executable, from entering the dispatcher to be dispatched and executed.

The data dependency hazards which occur with the simultaneous execution of multiple instructions in a single processor cycle have also been addressed by utilizing an approach known as "register renaming." Register renaming is a technique utilized to temporarily place the results of a particular instruction into a register for potential use by later instructions prior to the time the final result from an instruction is placed wi thin a register file. Register renaming is generally accomplished by providing a register file array with extra locations and a pointer arrangement to identify particular physical registers which have been assigned to logical registers. Selected prior art approaches also utilize multiple register file arrays to provide many "read" ports for data or for holding previous results for backup in the case of exceptions. While this technique provides the ability to simultaneously dispatch and execute multiple instructions where serial execution might otherwise be necessary, a problem exists with the dispatching of instructions to execution units utilizing such techniques. The requirement that an instruction utilize particular data or operands for execution has generally rendered it impossible to dispatch an instruction and associated data to an execution unit within a single

processor cycle since the dispatch unit, or the execution unit must generally perform a complex read through a lookup table or pointer system to determine when a temporary register contains the required data for execution of the instruction.

Furthermore, certain systems may rename a register more than once in a single processor cycle. Consequently, in those systems which utilize a lookup table for register renaming, a massive lookup table may be necessary in order to be able to determine the most recently assigned rename buffer for a particular register.

It should therefore be apparent that a need exist for a method and system which permits instruction dispatch within a superscalar processor system within a single processor cycle by permitting data or operands to be dispatched promptly with the instructions.

The above drawbacks of the prior art are overcome by the invention as claimed.

It is one object of the present invention to provide an improved data processing system.

It is another object of the present invention to provide an improved method and system for enhanced instruction dispatch efficiency in a superscalar processor system.

It is yet another object of the present invention to provide a method and system for indexing the assignment of intermediate storage buffers in a superscalar processor system.

The foregoing objects are achieved as is now described. A method and system is provided for enhanced instruction dispatch efficiency in a superscalar processor system having a plurality of intermediate storage buffers, a plurality of general purpose registers, and a storage buffer index. Multiple scalar instructions may be simultaneously dispatched from a dispatch buffer to a plurality of execution units. Each of the multiple scalar instructions generally include at least one source operand and one destination operand. A particular one of the plurality of intermediate storage buffers is assigned to a destination operand within a selected one of the multiple scalar instructions. A relationship between the particular one of the plurality of intermediate storage buffers and a designated one of the plurality of general purpose registers is stored in the storage buffer index at that time when the instruction which has been dispatched is replaced in the dispatcher by another instruction in the application program sequence. Results of execution from the selected one of the multiple scalar instructions are stored in the particular one of the intermediate storage buffers when the selected instruction is executed. The storage buffer index is used to determine which storage buffers to use as source operands for those instructions which are dispatched between the time that a storage buffer has been assigned for a specific general purpose register and the results of execution are moved from the storage buffer into the general purpose register.

The above as well as additional objects, features, and advantages of the present invention will become apparent in the following detailed written description.

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a high level block diagram of a superscalar processor system which may be utilized to implement the method and system of the present invention;

Figure 2 is a more detailed block diagram of the instruction dispatch circuitry of the superscalar processor system of Figure 1 which may be utilized to implement the method and system of the present invention;

Figure 3 is a high level block diagram which depicts the utilization of intermediate storage buffers within the superscalar processor system of Figure 1 in accordance with the method and system of the present invention;

Figure 4 is a pictorial representation of a storage buffer index within the superscalar processor system of Figure 3 which may be utilized to implement the method and system of the present invention; and

Figure 5 is a pictorial representation of an alternative storage buffer index within the superscalar processor system of Figure 3 which may be utilized to implement the method and system of the present invention.

With reference now to the figures and in particular with reference to Figure 1, there is depicted a high level block diagram of a superscalar data processing system 10 which may be utilized to implement the method and system of the present invention. As illustrated, data processing system 10 includes a memory 18 which is utilized to store data, instructions and the like. Data or instructions stored within memory 18 are preferably accessed utilizing cache/memory interface 20 in manner well known to those having skill in the art. The sizing and utilization of cache memory systems is a well known subspeciality within the data processing art and is not addressed within the present application. However, those skilled in the art will appreciate that by utilizing modern associative cache techniques, a large percentage of memory accesses may be achieved utilizing data temporarily stored within

cache/memory interface 20.

Instructions from cache/memory interface 20 are typically loaded into instruction dispatcher 22 which preferably includes a plurality of queue positions. In a typical embodiment of a superscalar data processing system each position within the instruction dispatch may include between two and eight instructions and thus, in a given cycle, up to eight instructions may be loaded into instruction dispatcher 22, depending upon how many valid instructions are passed by cache/memory interface 20 and how much space is available within instruction dispatcher 22.

As is typical in such superscalar data processing systems, instruction dispatcher 22 is utilized to dispatch instructions to execution units 24. As depicted within Figure 1, data processing system 10 may include one or more floating point units, fixed point processor units, load/store units, and a branch processor unit, as a matter of design choice. Thus, instruction dispatcher 22 may dispatch multiple instructions during a single cycle, one to each execution unit. Execution units may include reservation stations which could permit dispatch of more than one instruction to a single execution unit during a single cycle, as a matter of design choice. Thus, multiple execution units in a superscalar processor may each receive multiple instructions in a single cycle. Additionally, in multiprocessor systems instructions may be dispatched to multiple execution units 24 associated with multiple processors.

Referring now to Figure 2, there is depicted a more detailed block diagram of the instruction dispatch circuitry of the superscalar data processing system of Figure 1 which may be utilized to implement the method and system of the present invention. As illustrated, instructions from cache/memory interface 20 are typically loaded into instruction dispatcher 22 in groups in an application specified sequential order. Thus, as a matter of design choice, a group of two, four, or eight instructions are loaded from cache/memory interface 20 into instruction dispatcher 22, for dispatch on an opportunistic basis to one or more execution units within execution units 24. As depicted in the illustrated embodiment within Figure 2, these execution units may include a floating point execution unit 26, multiple load/store units 28 and 30, multiple fixed point execution units 32 and 34, and a branch execution unit 36. Of course, as discussed above, a greater or lesser number of execution units of different types may be included within data processing system 10 as a matter of design choice.

As is typical in superscalar data processing systems such as the type depicted herein, a plurality of general purpose registers are also provided. In the depicted embodiment within Figure 2, two groups of general purpose registers are provided. Floating point general purpose registers 44 and fixed point general purpose registers 46. Thus, the results of the execution of multiple instructions within the execution units depicted within Figure 2 are typically stored within a selected general purpose register for future utilization. In accordance with an important feature of the present invention, a plurality of intermediate storage buffers are also provided. That is, floating point intermediate storage buffers 40 and fixed point intermediate storage buffers 42. As will be explained in greater detail herein, each execution unit is connected to each intermediate storage buffer via an independent bus. Thus, data required by an execution unit, or generated by the execution of an instruction within an execution unit, may be placed on an independent bus and stored within a designated intermediate storage buffer for utilization by other execution units or for future transfer to a general purpose register.

In this manner, the maintaining of complex lookup tables typically associated with register renaming scheme is not required and an intermediate storage buffer may be efficiently assigned to be immediately utilized as a storage location for a destination operand, permitting the instruction to be dispatched within a single processor cycle, since data locations do not need to be determined within a renamed register file. Additionally, the execution of instructions in a non-sequential manner may be permitted, with the output of those instructions being temporarily stored within intermediate storage buffers for future assembly within the general purpose registers in the application specified sequential order.

As depicted, the fixed point execution units are each coupled via an independent bus to fixed point intermediate storage buffers 42, while the floating point execution unit is connected to each floating point intermediate storage buffer 40, via an independent bus. Load/store units 28 and 30, as those skilled in the art will appreciate, are necessarily connected to floating point intermediate storage buffers 40 and fixed point intermediate storage buffers 42, since these units will load and store both fixed point and floating point data.

In accordance with another important feature of the present invention a completion buffer 48 is provided. Completion buffer 48 is utilized to track the completion of the multiple scalar instructions which are being executed within execution units 24. The results of the execution of these instructions, as described above, are each temporarily stored within an associated intermediate storage buffer and, upon an indication that an instruction or a group of instructions have been completed successfully, in an application specified sequential or-

der, completion buffer 48 may be utilized to initiate the transfer of the results of those completed instructions data to the associated general purpose registers. Additionally, in the event an exception condition occurs, completion buffer 48 may be utilized to initiate storage of the results of the execution of any instruction which completed prior to the exception and to control instruction dispatcher 22 to restart the process at the point where the exception condition occurred, permitting corrupted or invalid data to be replaced by a valid output in an efficient manner, or to initiate a new sequence of instructions as specified by the application specific program for specific types of exceptions.

With reference now to Figure 3, there is depicted a high level block diagram illustrating the utilization of intermediate storage buffers within the superscalar processor system of Figure 1, in accordance with the method and system of the present invention. A storage buffer index 58 is provided which is accessible by instruction dispatcher 22. As will be described in greater detail herein, storage buffer index 58 is utilized to store and maintain an indication of the relationship between each of a plurality of intermediate storage buffers 60 and a designated general purpose register 62 for utilization during execution of succeeding instructions.

The plurality of intermediate storage buffers 60 are preferably connected to multiple independent buses 64. In the preferred embodiment, the number of independent buses 64 equals the number of intermediate storage buffers 60, with each intermediate storage buffer connected to one bus. Each of the intermediate storage buffers 60 may read from and write to a corresponding independent bus 64. Those skilled in the art will recognize that the number of buses utilized may vary as a matter of design choice.

A plurality of execution units 24a, 24b, 24c are coupled to each of the multiple independent buses 64. In the preferred embodiment, execution units 24a, 24b, 24c may include a floating point execution unit, multiple load/store units, multiple fixed point execution units, and a branch execution unit. However, those skilled in the art will appreciate that the number and type of execution units may vary as a matter of design choice.

Each of the plurality of execution units 24a, 24b, 24c may also read from and write to each of the independent buses 64. Consequently, each of the plurality of execution units 24a, 24b, 24c are coupled to each of the plurality of intermediate storage buffers 60 via the multiple independent buses 64. Thus, when data is generated by the execution of an instruction within an execution unit, the execution unit may place that data on a bus corresponding to a designated intermediate storage buffer which has been specified as a destination for that data, where the data may be temporarily stored. At the same time, the execution unit indicates the data on the bus is valid by setting a "valid" bit to an appropriate state. In this manner, while the data is on the bus, and before or at the same time the data is stored within the designated intermediate storage buffer, any other execution unit which requires that data may retrieve the data from the bus. Thus, one advantage of the provision of independent buses is the elimination of the need to store the data in a buffer and then thereafter access that data from the buffer. The ability to retrieve data directly from the bus will significantly increase the operation speed of the processor system.

Still referring to Figure 3, it may be seen that multiple independent buses 64 are each coupled to general purpose registers 62. When an instruction is to be dispatched to an execution unit, the relationship between the designated intermediate storage buffer and the selected general purpose register is preferably stored within the storage buffer index 58. When the results of execution are needed by another execution unit, the transfer of those results may be performed utilizing the information within storage buffer index 58. As will be discussed in greater detail herein, the information stored within storage buffer index 58 may be an identifier for the designated intermediate storage buffer which is accessed utilizing the general purpose register identification or an identifier of a general purpose register which is accessed with the identification of the designated intermediate storage buffer.

In this manner, the maintenance of complex lookup tables typically associated with a register renaming scheme is not required, since an intermediate storage buffer may be immediately assigned as a storage location for each destination operand within an instruction to be dispatched, without requiring the renaming of a general purpose register file. Furthermore, storage buffer index 58 permits the execution of instructions in a nonsequential manner, since the results of execution of each instruction are temporarily stored in intermediate storage buffers 60, and may be subsequently utilized by a succeeding instruction by utilizing the information within storage buffer index 58 and the content of completion buffer 48 (see Figure 2).

Figure 4 is a pictorial representation of a storage buffer index within the superscalar processor system of Figure 3 which may be utilized to implement the method and system of the present invention. Storage buffer index 58 comprises a number of entries, numbered 0-r. R equals the number of general purpose registers in the superscalar processor system. As discussed above, when an in-

struction which has previously been dispatched to an execution unit is replaced in the dispatcher by a subsequent application program specific instruction, the relationship between the destination operand within the dispatched instruction and the designated intermediate storage buffer is store in storage buffer index 58. In this embodiment, an identifier for the designated intermediate storage buffer is stored in the entry corresponding to a selected general purpose register.

For example, in entry position 0, which corresponds to general purpose register 0, an identifier for intermediate storage buffer 3 has been stored. Additionally, in entry position r, which corresponds to general purpose register r, an identifier for intermediate storage buffer 2 has been stored. Thus, when the results of execution are to be transferred from intermediate storage buffer 3 to another execution unit, the transfer is accomplished by utilizing entry 0 in storage buffer index 58.

Figure 5 is a pictorial representation of an alternative storage buffer index within the superscalar processor system of Figure 3 which may be utilized to implement the method and system of the present invention. Storage buffer index 58 comprises a number of entries, numbered 0-n. N equals the number of intermediate storage buffers in the superscalar processor system. When an instruction is dispatched to an execution unit, an identifier for the selected general purpose register is stored in the entry corresponding to the designated intermediate storage buffer where the results of execution are to be stored.

For example, in entry position 0, which corresponds to intermediate storage buffer 0, an identifier for general purpose register 25 has been stored. Additionally, in entry position 2, which corresponds to intermediate storage buffer 2, an identifier for general purpose register 2 has been stored. Thus, when the results of execution are to be transferred from intermediate storage buffer 0 to an alternate execution unit, the transfer is coordinated by utilizing the entry 0 in storage buffer index 58.

Upon reference to the foregoing, those skilled in the art will appreciate that a method and system have been provided for indexing the assignment of intermediate storage buffers in a superscalar processor system. The need for massive associative lookup tables is eliminated, since the results of execution are stored temporarily in intermediate storage buffers until such time the results are stored in general purpose registers in an application specified order.

Claims

1. A method for enhanced instruction dispatch efficiency in a superscalar processor system (10) capable of simultaneously dispatching multiple scalar instructions from a dispatch buffer (22) to a plurality of execution units (24), each of said multiple scalar instructions including at least one source operand and one destination operand, wherein results of execution of said multiple scalar instructions are stored in a plurality of general purpose registers (62), said method comprising the steps of:

   assigning a particular one of a plurality of intermediate storage buffers (60) to a destination operand within a selected one of said multiple scalar instructions;

   storing a relationship between said particular one of said plurality of intermediate storage buffers (60) and a designated one of said plurality of general purpose registers (62) in a storage buffer index (58);

   storing results of execution of said selected one of said multiple scalar instructions in said particular one of said plurality of intermediate storage buffers (60); and

   transferring said results of execution from a particular one of said plurality of intermediate storage buffers (60) to an alternate one of said plurality of execution units (24) for utilization in execution of an alternate one of said multiple scalar instructions utilizing said storage buffer index (58).

2. The method for enhanced instruction dispatch efficiency in a superscalar processor system (10) of Claim 1, wherein said step of storing a relationship between said particular one of said plurality of intermediate storage buffers (60) and a designated one of a plurality of general processing registers (62) in a storage buffer index (58) comprises storing an identifier for said particular one of said plurality of intermediate storage buffers (60) in said storage buffer index (58), and accessing said identifier for said particular one of said plurality of intermediate storage buffers (60) from said storage buffer index (58) utilizing an identifier for said designated one of a plurality of general processing registers (62).

3. The method for enhanced instruction dispatch efficiency in a superscalar processor system (10) of Claim 1, wherein said step of storing a relationship between said particular one of said plurality of intermediate storage buffers (60) and a designated one of a plurality of general processing registers (62) in a storage buffer

index (58) comprises storing an identifier for said designated one of said plurality of general processing registers (62) in said storage buffer index (58), and accessing said identifier for said designated one of said plurality of general processing registers (62) from said storage buffer index (58) utilizing an identifier for said particular one of said plurality of intermediate storage buffers (60).

4. A system for enhanced instruction dispatch efficiency in a superscalar processor system (10) capable of simultaneously dispatching multiple scalar instructions from a dispatch buffer (22) to a plurality of execution units (24), each of said multiple scalar instructions including at least one source operand and one destination operand, wherein results of execution of said multiple scalar instructions are stored in a plurality of general purpose registers (62), said system comprising:

means for assigning a particular one of a plurality of intermediate storage buffers (60) to a destination operand within a selected one of said multiple scalar instructions;

means for storing a relationship between said particular one of said plurality of intermediate storage buffers (60) and a designated one of said plurality of general purpose registers (62) in a storage buffer index (58);

means for storing results of execution of said selected one of said multiple scalar instructions in said particular one of said plurality of intermediate storage buffers (60); and

means for transferring said results of execution from a particular one of said plurality of intermediate storage buffers (60) to an alternate one of said plurality of execution units (24) for utilization in execution of an alternate one of said multiple scalar instructions utilizing said storage buffer index (58).

5. The system for enhanced instruction dispatch efficiency in a superscalar processor system (10) of Claim 4, wherein said means for storing a relationship between said particular one of said plurality of intermediate storage buffers (60) and a designated one of a plurality of general processing registers (62) in a storage buffer index (58) comprises means for storing an identifier for said particular one of said plurality of intermediate storage buffers (60) in said storage buffer index (58), and means for accessing said identifier for said particular one of said plurality of intermediate storage buffers (60) from said storage buffer index (58) utilizing an identifier for said designated one of a plurality of general processing registers (62).

6. The system for enhanced instruction dispatch efficiency in a superscalar processor system (10) of Claim 4, wherein said means for storing a relationship between said particular one of said plurality of intermediate storage buffers (60) and a designated one of a plurality of general processing registers (62) in a storage buffer index (58) comprises means for storing an identifier for said designated one of said plurality of general processing registers (62) in said storage buffer index (58), and means for accessing said identifier for said designated one of said plurality of general processing registers (62) from said storage buffer index (58) utilizing an identifier for said particular one of said plurality of intermediate storage buffers (60).

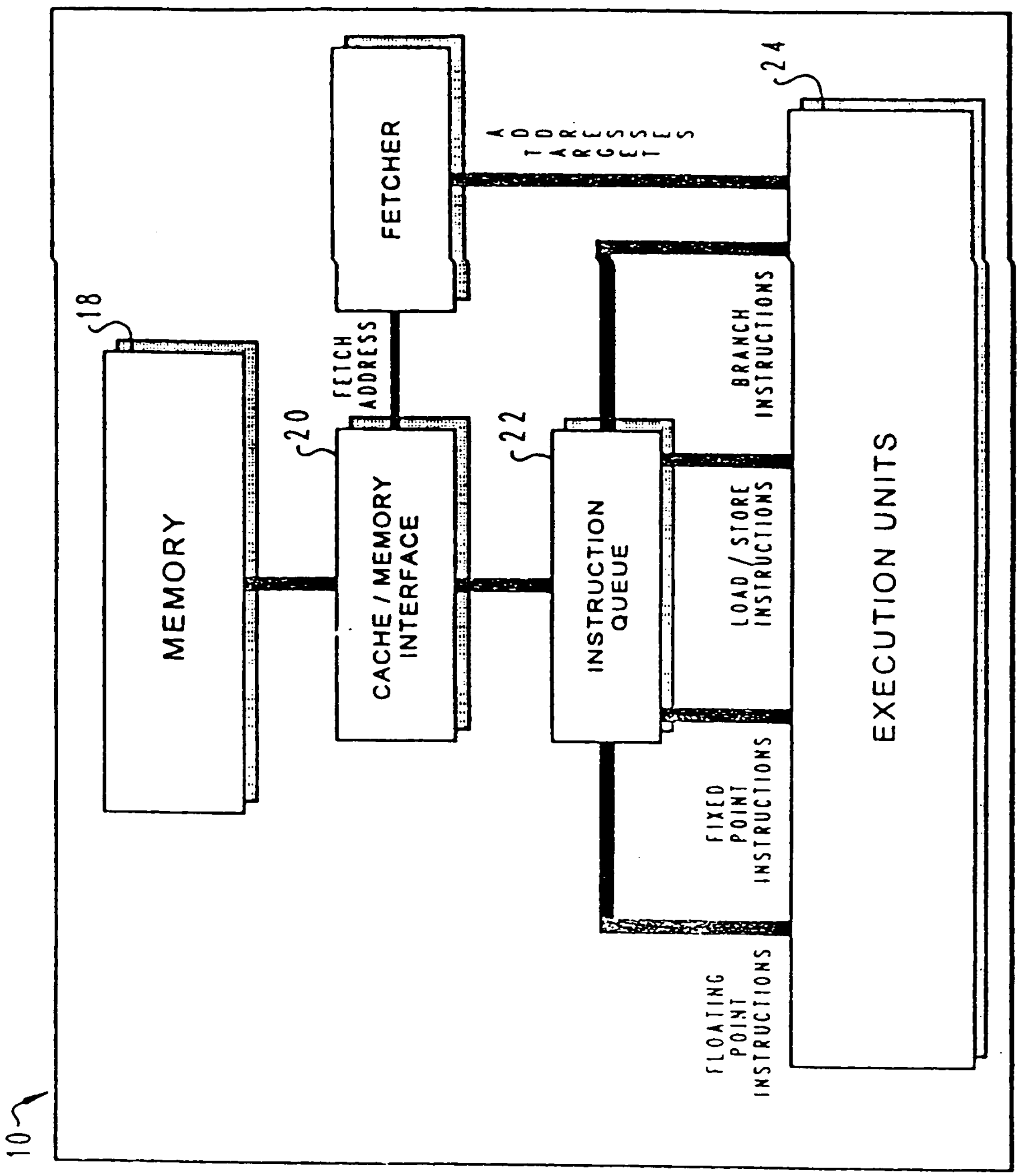
10
MEMORY
18
CACHE / MEMORY INTERFACE
20
FETCH ADDRESS
FETCHER
ADDRESSES
TARGETS
INSTRUCTION QUEUE
22
BRANCH INSTRUCTIONS
LOAD/STORE INSTRUCTIONS
FIXED POINT INSTRUCTIONS
FLOATING POINT INSTRUCTIONS
EXECUTION UNITS
24

Fig. 1

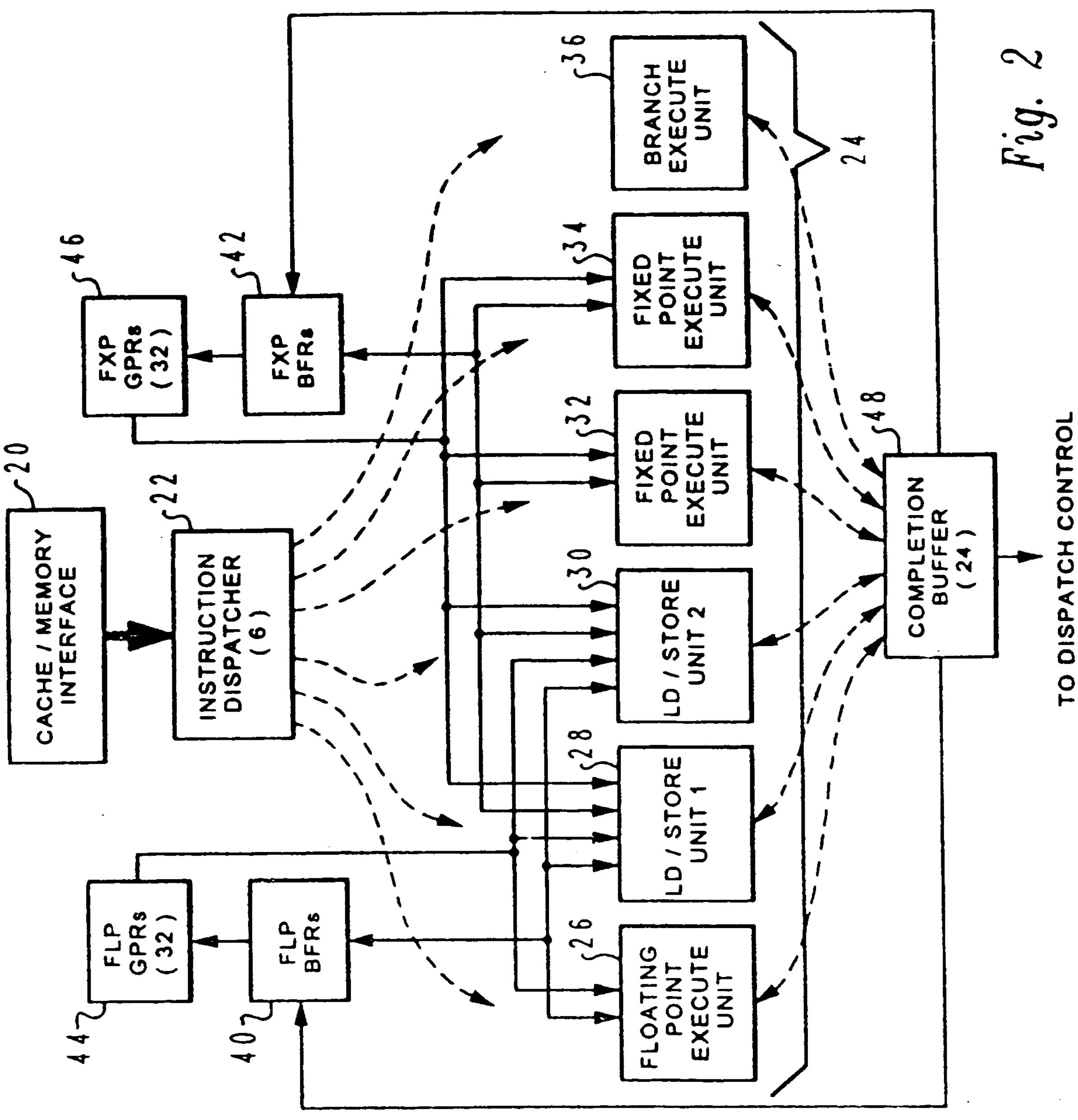
CACHE / MEMORY INTERFACE
20
INSTRUCTION DISPATCHER (6)
22
FXP GPRs (32)
46
FXP BFRs
42
FLP GPRs (32)
44
FLP BFRs
40
FLOATING POINT EXECUTE UNIT
26
LD / STORE UNIT 1
28
LD / STORE UNIT 2
30
FIXED POINT EXECUTE UNIT
32
FIXED POINT EXECUTE UNIT
34
BRANCH EXECUTE UNIT
36
24
COMPLETION BUFFER (24)
48
TO DISPATCH CONTROL

Fig. 2

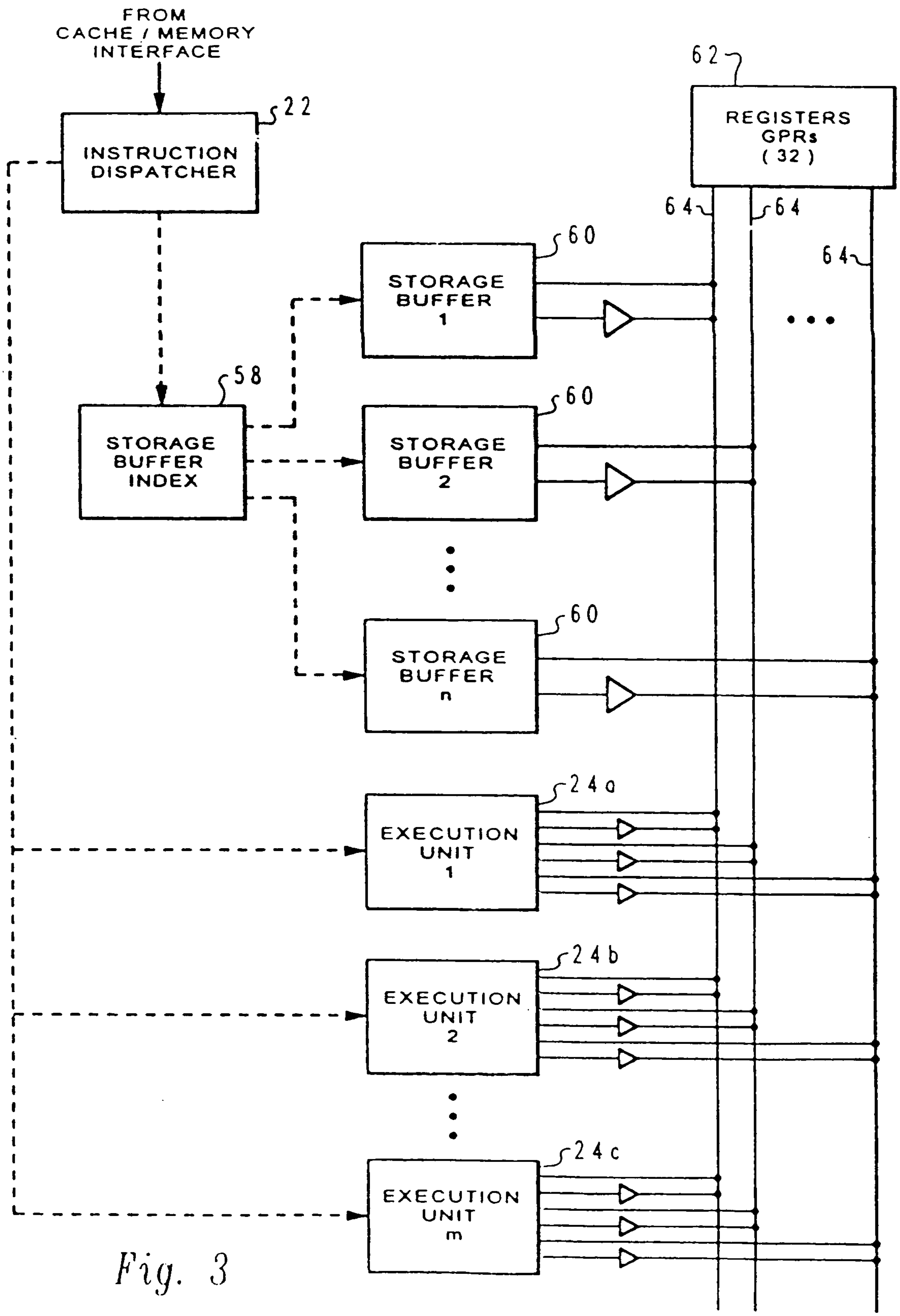
FROM
CACHE / MEMORY
INTERFACE
22
INSTRUCTION
DISPATCHER
62
REGISTERS
GPRs
(32)
64
64
64
60
STORAGE
BUFFER
1
58
STORAGE
BUFFER
INDEX
60
STORAGE
BUFFER
2
60
STORAGE
BUFFER
n
24a
EXECUTION
UNIT
1
24b
EXECUTION
UNIT
2
24c
EXECUTION
UNIT
m

Fig. 3

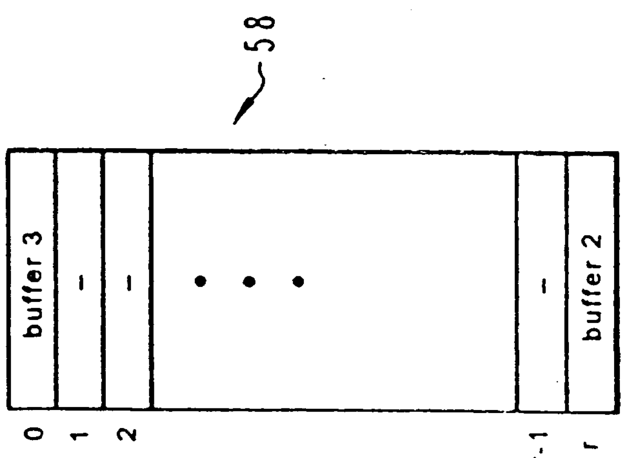

Fig. 4

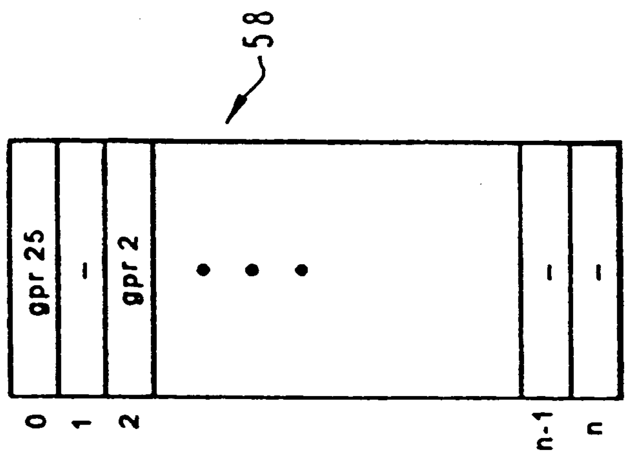

Fig. 5

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 93 12 0933

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
X	MIKE JOHNSON 'Superscalar Microprocessor Design' 1991 , PRENTICE HALL , ENGLEWOOD CLIFFS, NEW JERSEY, US. see page 44, section 3.4: 'The Design of the Standard Processor' see page 168, section 9.1.2: 'Instruction Decode' -----	1-6	G06F9/38
			TECHNICAL FIELDS SEARCHED (Int.Cl.5)
			G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	31 March 1994	Daskalakis, T

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)